VALIDATION OF SPACE/GROUND ANTENNA CONTROL ALGORITHMS USING A COMPUTER-AIDED DESIGN TOOL

Final Report
NASA/ASEE Summer Faculty Fellowship Program -- 1994
Johnson Space Center

Prepared by:	Dr. Rex E. Gantenbein
Academic rank:	Associate Professor
College and department:	University of Wyoming Dept. of Computer Science Laramie WY 82071-3682
NASA/JSC	
Directorate:	Engineering
Division:	Tracking and Communications
Branch:	Electromagnetic Systems
JSC colleague:	Andrew Benjamin
Date submitted:	5 August 1994
Contract number:	NGT-44-005-803

ABSTRACT

The validation of the algorithms for controlling the space-to-ground antenna subsystem for Space Station Alpha is an important step in assuring reliable communications. These algorithms have been developed and tested using a simulation environment based on a computer-aided design tool that can provide a time-based execution framework with variable environmental parameters.

Our work this summer has involved the exploration of this environment and the documentation of the procedures used to validate these algorithms. We have installed a variety of tools in a laboratory of the Tracking and Communications Division for reproducing the simulation experiments carried out on these algorithms to verify that they do meet their requirements for controlling the antenna systems. In this report, we describe the processes used in these simulations and our work in validating the tests used.

INTRODUCTION

Designing algorithms for real-time control of communications devices is difficult, but even more so is validating the correctness of that design. The consequences of design errors are high in many applications of this type, especially where manned missions are dependent on the system. The users of such systems are often justifiably concerned about the possibility of design errors being propagated through the development process into the completed and deployed system. Design errors can be a significant problem in making computer-driven control systems dependable. In many cases, the manifestations of design faults become noticeable only after implementation, at which point correction can be expensive due to the amount of rework that may be required to eliminate the problem.

One approach proposed to combat this problem is validation of a computer system design prior to implementation. Design validation can improve the quality and reduce the cost of a computer system by eliminating design faults before implementation begins. Since a design cannot be "tested" in the usual manner, other techniques based on formal logic or simulation must be employed to assure both the developers and the users that the system, as designed, will behave according to their expectations and needs.

In this summer's project, the author has explored ways to validate the algorithms developed for controlling the Space-to-Ground Antenna (SGANT) subsystem to be used on board Space Station Alpha for communicating with the Tracking and Data Relay Satellite (TDRS) system. These designs are very complex and thus prone to design faults that might be overlooked using traditional testing methods. The goal of this work is to determine effective ways of validating these designs prior to their implementation to assure the highest quality systems at all stages of the development process.

PROJECT OVERVIEW

The SGANT system is being designed by the Satellite and Communications Systems Division of SPAR Aerospace Ltd. of Quebec, Canada. Dynacon Enterprises Ltd., in Ontario, Canada, is designing three control algorithms for this system under contract to SPAR. The three algorithms control slew, search, and pull-in/tracking for the SGANT system.

To verify the performance of these algorithms and demonstrate their conformance to the requirements specified by

SPAR, Dynacon developed two simulation facilities. These are the Final Verification Simulator (FVS) and the Probability of Successful Search Simulator (PSS) [Dynacon92]. The FVS is a collection of simulations based on a set of common subsystem models and represents a comprehensive model of the SGANT system. The PSS simulator addresses the statistical performance of the TDRS search and acquisition functions in SGANT.

The FVS simulator includes a discrete-time simulation of the control algorithms and body dynamics of the system as deployed both in micro-gravity and in full gravity (for evaluation in a ground-based lab). This simulator produces a time-based history of the algorithm behavior under various parameters. This history is used as input to the PSS simulator, which executes a series of searches over this history to determine the likelihood of successful acquisition of the TDRS signal. Together, the two simulators provide a means of verifying that the control algorithms, as designed, are capable of meeting the required 90% or more probability of a successful search.[1]

VALIDATION ENVIRONMENT

The goal of the summer project was to recreate the environment used by Dynacon to validate the SGANT control algorithms. This required the use of a variety of different tools, which were installed on the SUN Sparcstation (**commlab**) housed in the Communications Laboratory in Building 14 at JSC. In this section we describe briefly the tools used.

MATRIX$_X$

The FVS was implemented using the SystemBuild software of MATRIX$_X$, an automated design tool that provides graphic representation of a variety of system building blocks as well as simulation by discrete or continuous time execution is [ISI92]. This engineering system design tool supports the development of graphics-based designs that can be executed on data tables that represent sensor inputs, as well as a hierarchy of components that can be packaged into "super-

[1]As pointed out in [Dynacon92], the probability of successful search is an estimate of the probability of acquisition, which is based on the ability of the system both to locate the TDRS target and to pull it in and track it. The computation required to simulate the second probability is, according to the documentation, "too prohibitive at this time to allow a statistically significant estimate" of acquisition.

blocks" in the design to provide composability and incremental development. It also allows user code blocks in programming languages (such as FORTRAN) to be incorporated into a simulation. This feature was used to insert the control algorithms, as implemented, into the simulation for validation.

GNU C compiler

The PSS simulator was written in the C programming language, in conformance with the ANSI C standard. The C compiler provided with the SUN operating system on commlab was not an ANSI compiler, however, but conformed to the older standard of C. Normally, there are some syntactic differences between ANSI and non-ANSI C that can be easily resolved, but in this particular instance, conversion of the PSS programs to non-ANSI (SUN-compatible) C code resulted in the program producing incompatible output.

In an attempt to quickly resolve this conflict, we installed version 2.6.0 of the GNU C compiler, which we retrieved by anonymous FTP from Massachusetts Institute of Technology. This compiler conforms to the ANSI standard syntax and, after installation, was used to compile the PSS programs on commlab. The procedures for retrieving and installing GNU C are included as Appendix A.

As documented later in the report, this compiler was still not able to reproduce the PSS behavior exactly. For this reason, a full, commercial ANSI C compiler was purchased and installed in August 1994. Testing of this compiler's results against those from Dynacon is still being completed.

GNUPLOT

Since the data sets produced by the FVS and PSS simulators are so large (depending on the duration of the simulated run and the size of the time frame, as will be described), it is necessary to have some plotting tools available to look at and summarize the data. The $MATRIX_x$ simulations provide plots of selected output data, but it was inconvenient to invoke this package for a simple plot of data. For this reason, we installed another package available by anonymous FTP from M.I.T., GNUPLOT. This package is not part of the GNU project as is GNU C, but provides a simple environment for producing two- and three-dimensional plots. We found this facility easy to use and very handy for quick analysis of data during this project. The procedures for retrieving, installing, and running GNUPLOT are included as Appendix B.

VALIDATION PROCEDURES

What we attempted to do in this project was to validate the results reported in [Dynacon92] by recreating the simulations in the Communications Laboratory. This involved the following steps:

- running the FVS simulations under $MATRIX_x$ to produce the time history for the tracking and crosstracking error;
- extracting the tracking and crosstracking data into a file that can be used by the PSS simulator;
- running the PSS simulator on the tracking/crosstracking data file; and
- analyzing the results.

In this section, we describe the procedures needed to recreate the simulations. A script for performing simulations is included as Appendix C.

Running the FVS simulation in $MATRIX_x$

The FVS simulator is driven by batch runscripts executed from a custom $MATRIX_x$ executable file. This executable is currently installed as "fvsim_course3.1" in the directory **/usr2/home/designlab/ben/course** on commlab.[2] When new versions of $MATRIX_x$ are installed, or if this file becomes corrupted, it will be necessary to install it. The instructions for making a new custom executable are included as Appendix D.

The "fvsim_course3.1" file is executed to invoke the $MATRIX_x$ core. At the "<>" prompt, the batch runscript for the simulation desired can be started with the $MATRIX_x$ core **exec** command. The runscripts used in the Dynacon simulations are documented in Section 4 ("Algorithm Verification") of the final design report [Dynacon92]. There are several parameters in the runscripts that can change the FVS simulation and affect the results. The most important of these are listed in Table I below.

The runscripts add these parameters and other data values

---

[2]This directory is the "home" directory for the simulation process. Throughout this document, we will use a "." to refer to this directory when discussing file structures.

Table I. $MATRIX_x$ runscript parameters.

VARIABLE	VALUE	DESCRIPTION
choice	1,2,3,4,5	select algorithms to run (typically 1)
serial	string	used to name output files
totaltime	integer	simulation time (seconds)
DToutput	decimal	time frame (seconds)
autotrack	0, 1	autotracking simulation: off (0) / on (1)
slew	0, 1	slew simulation: off (0) / on (1)
search	0, 1	TDRS search simulation: off (0) / on (1)
track	0, 1	antenna track simulation: off (0) / on (1)
TDRS	0, 1	custom (0) TDRS motion or start at zenith (1)
Sr	3.5, 26.5	TDRS mean signal level (should match the level used in PSS simulation)
CMSON	0, 1	CMS Estimate noise: off (0) / on (1)
x/y/zrot	decimal	rotation disturbances
x/y/ztran	decimal	translation disturbances
seed1/2/3/4	integer	set CMS position and velocity noise
do_sim<x>	_big, _medium, _slew, _slew2, 2, <blank>	select runscript to execute ("define" at start of the batch file must provide pathname for the runscript)

for the FVS simulation to the MATRIX$_x$ data stack, then invoke another runscript to actually perform the simulation. These runscripts are stored in the directory **./udc_files** with the ".udc" extension. Some additional files are also included in this directory that are invoked from the udc runscripts.

Depending on the runscript invoked (and the size of the ensuing simulation), the time required for the simulation can run from just under two hours to approximately 48 hours (for a full simulation). It is advisable to consider specifically what parameters are needed prior to execution. (In Appendix C, we describe the runscripts and parameters needed to prepare data for the PSS simulator.)

Extracting the data for the PSS simulator

Once the FVS simulation has completed (messages describing the percentage of completion will appear; when the simulation has reached 100%, the "<>" prompt will be displayed), a number of variables will have been created by the simulation. These variables are stored in the N-by-57 matrix "y" on the MATRIX$_x$ stack, where N is determined by the time duration parameter "totaltime" and the time frame parameter "DToutput" in the batch runscript. In addition, this data has been saved in a file in the **./output** directory, under the file name "out_<serial>" where <serial> is also defined in the runscript. Data from the FVS simulation can be recovered from these file by invoking the MATRIX$_x$ core and loading this file.

Each row of this matrix represents the values of a variable computed at each time frame in the simulation. Of particular interest for the PSS simulation are rows 15 and 16, which contain the values for the track (TK) and cross-track (XTK) boresight tracking angle error created by the search simulation in the FVS. The 35-second spiral search portions of these two vectors should be saved (as an ASCII file, using the MATRIX$_x$ command "fsave") in a separate file for input to the PSS simulator.

This file will contain the values of TK and XTK as data pairs. However, MATRIX$_x$ may not always format the file in two columns, so it is necessary to pass this file through a "filter" program, written by Dynacon, prior to its input to the PSS program. This is a C program that simply reads in the data and writes every third data pair, starting with the first, to standard output, formatting them into two columns. NOTE: MATRIX$_x$ also attaches some labeling information to the front of the saved file (typically three lines of text). The current filter program will not accept these three lines, so

the file must be edited by hand prior to running the filter program. Appendix E contains C code that could be added to the filter program to automatically skip over these lines.

Running the PSS simulation

The TK/XTK data from the verifications simulations performed by Dynacon reside as files with the name "pss_<n>" in the directory **./PSS/csim**, where <n> is the simulation run number as described in [Dynacon92]. This directory is also where the filter program resides as "filter.c" for the source code and "filter" for the executable.

Filtering the "pss_<n>" files produces a spiral data file that represents the time history of the simulation for tracking TDRS. Each of the spiral data files has the name "spiraldat<n>" to correspond with the files from which it was produced. These files are used as input to the PSS simulator.

The PSS simulator is composed of four separate C source files: "input.c", "poa.c", "sim.c", and "pwrl.c". Prior to running a PSS simulation on a spiral data file, it is necessary to modify "input.c" to set the number of simulation runs, the spiral data set number <n>, and the mean TDRS signal level (low 3.5 dB, high 26.5 dB). These values should be assigned to the variables "n_runs", "spiral", and "mean", respectively.

In addition, the simulation uses the C random number generator to help determine the probabilistic behavior of the search. In order to exactly recreate the results of a particular Dynacon simulation, the seed for the random number generator (which normally is set to the process id of the executing simulation), must be initialized to the same value as the Dynacon run. This is accomplished by editing the file "poa.c" and setting the variable "pid" to the value used in the original run. This value is recorded in the outputs of the Dynacon simulations in **./PSS/csim** under the file names "pss_<n>_<level>.plot", where <n> is the spiral data set number and <level> is "low" or "high" corresponding to the mean TDRS signal level used in the simulation.

Analyzing the outputs of the PSS simulation

The output from the Dynacon PSS simulations is summarized in Appendix F for the corresponding PSS simulation run number, spiral data set number and signal level. (Also included is the random number generator seed used for each simulation, as described in the previous section.) The last column in the table in the Appendix represents the probability of successful

search for a maximum pull-in radius of 0.6 degrees. The specifications call for this value to be 90% or greater.

The actual output from the PSS simulator contains a range of probabilities for pull-in radius values from 0.1 to 0.8 degrees. Obviously, as the radius increases, the probability of successful search will increase. The results were plotted by Dynacon using the XPLOT tool, producing postscript-formatted files stored as "pss_<n>_<level>.ps" in the **./PSS/csim** directory. They can also be plotted using GNUPLOT. (NOTE: the output files contain a header line describing the spiral data set number, signal level, number of runs, and seed. This line must be manually removed prior to plotting with GNUPLOT.) Examples of these output files and the associated plots are shown in Appendix G.

EVALUATION OF EXPERIENCES

It is difficult to say exactly how well we were able to verify the results of the FVS/PSS simulation during this summer's work. Part of the problem was that we were unable to collect data from Dynacon until well into July, so that our time to work with their simulations and data was effectively halved.

Once we had loaded the information from Dynacon onto commlab (which, incidentally, involved installing additional disk space to hold the data and provide room to run the simulations), our first work was to try to run the PSS simulator on the spiral data sets provided by Dynacon. The PSS simulator was compiled under an ANSI C compiler at Dynacon that was not available on commlab. It was fairly simple to convert the syntax of the PSS code from ANSI to "old style" C, but when we ran the simulator, the probabilities produced were always zero! After some investigation, we concluded that the reason for this was in the way the compilers handled random number generation. To a degree, this was verified when we retrieved and installed GNU C (which conforms to the ANSI standard syntax) and ran the original programs. The probabilities produced were non-zero and agreed (for simulations at the low signal level, at least) with those documented by Dynacon. We are still investigating the discrepancies in the high signal level results.

One ongoing problem was the use of the constant "RAND_MAX" in the PSS simulator code. This constant, which represents the maximum integer to be generated by the random number facility, is defined as one less than the maximum integer by the ANSI standard (so, on commlab, this value would

be $2^{31}-2$, or 2,147,483,646). The constant was not part of the GNU C environment, so it was inserted into the source code in the file "sim.c". Since the low-level signal simulations seemed to be reproducible, we believe that this works, but if an ANSI C compiler is installed, this definition should be removed.

Once we had addressed the problems with the PSS simulator, we turned to the FVS simulator and attempted to see how to create the data files that the PSS simulator used as input. The documentation on this process included in the report was incomplete; so a number of "problem reports" had to be created for Dynacon to fill in the gaps. The procedure used for sending these reports to Dynacon over Internet is included as Appendix H. Eventually, these procedures were collected together (see Appendix C) so that we could attempt to recreate the simulations.

At this time, we believe we understand the procedures by which the simulations were run, but we have not yet been able to completely recreate and validate the results. We expect that this project will require more time and effort. Among the lessons that can be learned from this project is the value of good organization in documenting a project. It may be that the documents provided by Dynacon were not meant to be a "user's manual" for the simulation system but rather a demonstration of the algorithms' compliance with their requirements through unplanned (some would prefer "artistic") testing. There is no indication of a test plan included in the documentation available here, so it is not at all clear that such a plan exists. While there is little reason to doubt that the testing was done and that the algorithms do indeed meet their requirements, there is little evidence at this time that unequivocally demonstrates that they do so.

REFERENCES

[Dynacon92] SSF Space/Ground Antenna Control Algorithms: Final Design Report, Volume II: Validation and Verification. Part A -- Simulator Description and Validation, Algorithm Verification Overview. Dynacon Enterprises Limited, 1992.

[ISI92] SystemBuild/WS: User's Guide for Version 3.0, Part 000-0051-002. Integrated Systems, Inc., 1992.